* Review design feedback [10 min]
  + Data model: how to indicate a user’s average rating is an aggregate of the ratings from their reviews
  + Why do we need an object that encompasses both location and time?
  + Insights?
* Go over what we have so far [10 min]
  + Model structure + methods
  + Models with JS objects in addition to mongoose, or just mongoose?
* Next steps [10 min]
  + Routes

Talk about the comments provided in email

* + D: group location and time together so that a ride has details and those encompass location/ time/ other information. It’s also fine the way it is. The way we have it, a location has a ride, which means you can’t have future rides to that same location. <- we fixed this by adding immutability marking!
  + B: We wanted to encourage commitment to the ride with immutable location/ time per ride
  + D: Talking about Kerberos…. Certificates? Not sure how feasible, might not be worth time to research certificate authentication. Just verify that people are registering with MIT emails.
  + T: how to we indicate the average rating on the data model, as related to ratings which are attached to reviews.
  + D: write a textual constraint: a user also has an average rating associated with it, which is an aggregate of the user’s review’s ratings. Will you be allowing the reviewer to edit the reviews? <- We decided that they will be allowed to change the content of the review, but the review will always be attached to the User. Good insight: this stuff^.   
    A ride has a total capacity, which has properties that of maximum and current -> cleaner data model   
    another insight: when a ride has 0 remaining riders, it will delete itself  
    Kerberos is not necessary probably. Just a general User would be ok. But it’s up to us in our actual implementation.
* What we’ve done on the models so far:
  + D: You can create your model with mongoose, forces each collection to have a certain structure. Also allows you to perform SQL like queries. Allows us to do joints. Not necessary!
* On routes:
  + a body parameter saying “add myself” or “remove myself” The URL will be the same, however it’s like a flag saying add or delete.
  + /users/:id/review/?ride=<id>  
    body, path(URL), query(after the ?)
  + to modify, if you post a new review/ edit, override the previous review
  + if we have a variable like body.content OR content, it’ll take whatever is in content if there is no body.content

next steps:

* + aiming to do routes, UI, frontend by Friday.
  + The MVP is being able to create rides! Review stuff isn’t necessary.
* Other notes:
  + Callback format is usually callback(error, data) just so that it is easier to match. Ignore the first parameter if there is an error. Don’t only check for null because it could have an error and things would break
  + Error would almost never trigger for user exists. Forming a query will probably not.
  + Rides might be an empty array if we find nothing/ collection is empty.
  + Use update and addtoset so that there will be no duplicates. Use timestamps, default = date.time.now so that the sets will have some order
  + Sorting by riderID may not necessarily mean it is time sorted, since IDs could be randomly generated
* Use the **Moment.js** library if we want to use / organize/ compare dates & times